Chapter 4 : Intermediate SQL

Outline

- Join Expressions
- Views
- Transactions
- Integrity Constraints
- SQL Data Types and Schemas
- Authorization

Integrity Constraints

■ Integrity constraints guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.

- A checking account must have a balance greater than $10,000.00
- A salary of a bank employee must be at least $4.00 an hour
- A customer must have a (non-null) phone number

Constraints on a Single Relation

- **not null**
- **primary key**
- **unique**
- **check** (P), where P is a predicate

Not Null Constraints

- **not null**
  - Declare *name* and *budget* to be **not null**

    name **varchar**(20) **not null**
    budget **numeric**(12,2) **not null**

Unique Constraints

■ **unique** (A_1, A_2, …, A_m)

- The unique specification states that the attributes A_1, A_2, …, A_m form a candidate key.

- Candidate keys are permitted to be null (in contrast to primary keys).

The check clause

■ The **check** (P) clause specifies a predicate P that must be satisfied by every tuple in a relation.

■ Example: ensure that semester is one of fall, winter, spring or summer

```
create table section
    (course_id varchar (8),
     sec_id varchar (8),
     semester varchar (6),
     year numeric (4,0),
     building varchar (15),
     room_number varchar (7),
     time slot id varchar (4),
     primary key (course_id, sec_id, semester, year),
     check (semester in ('Fall', 'Winter', 'Spring', 'Summer')))
```

Referential Integrity

- Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.

    - Example: If "Biology" is a department name appearing in one of the tuples in the *instructor* relation, then there exists a tuple in the *department* relation for "Biology".

- Let A be a set of attributes. Let R and S be two relations that contain attributes A and where A is the primary key of S. A is said to be a **foreign key** of R if for any values of A appearing in R these values also appear in S.

Referential Integrity (Cont.)

- Foreign *keys can be* specified as part of the SQL **create table** statement

  foreign key (*dept_name*) **references** *department*

- By default, a foreign key references the primary-key attributes of the referenced table.

- SQL allows a list of attributes of the referenced relation to be specified explicitly.

  foreign key (*dept_name*) **references** *department* (*dept_name*)

Cascading Actions in Referential Integrity

- When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

- An alternative, in case of delete or update is to cascade

  create table *course* (

      (…
      dept_name **varchar**(20),
      foreign key (*dept_name*) **references** *department*
          on delete cascade
          on update cascade,
      . . .)

- Instead of cascade we can use :

  - **set null**,
  - **set default**

Integrity Constraint Violation During Transactions

- Consider:

  create table *person* (
      ID **char**(10),
    name **char**(40),
    mother **char**(10),
    father **char**(10),
    primary key *ID,*
    foreign key *father* **references** *person,*
    foreign key *mother* **references** *person*)

- How to insert a tuple without causing constraint violation?

  - Insert father and mother of a person before inserting person

  - OR, set father and mother to null initially, update after inserting all persons (not possible if father and mother attributes declared to be **not null**)

  - OR defer constraint checking

Complex Check Conditions

■ The predicate in the check clause can be an arbitrary predicate that can include a subquery.

check (*time_slot_id* **in** (**select** *time_slot_id* **from** *time_slot*))

The check condition states that the time_slot_id in each tuple in the *section* relation is actually the identifier of a time slot in the *time_slot* relation.

● The condition has to be checked not only when a tuple is inserted or modified in *section* , but also when the relation *time_slot* changes

Built-in Data Types in SQL

- **date:** Dates, containing a (4 digit) year, month and date
  - Example: **date** '2005-7-27'
- **time:** Time of day, in hours, minutes and seconds.
  - Example: **time** '09:00:30'   **time** '09:00:30.75'
- **timestamp:** date plus time of day
  - Example: **timestamp** '2005-7-27 09:00:30.75'
- **interval:** period of time
  - Example: interval '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values

Large-Object Types

- Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:

    - **blob**: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)

    - **clob**: character large object -- object is a large collection of character data

- When a query returns a large object, a pointer is returned rather than the large object itself.